Wireshark Packet Analysis Project Report

Project Title:

Real-Time Network Traffic Analysis Using Wireshark

Objective:

To analyse real-time network traffic using **Wireshark** to detect suspicious activities such as DNS abuse, TCP retransmissions, brute-force attempts, and connection resets.

Tools Used:

- Wireshark

- Npcap

- Windows 11 / Kali Linux

Methodology:

1. Setup & Capture

Installed and launched Wireshark with admin privileges.

Selected the active network interface (Wi-Fi).

Captured live traffic for 5 minutes while browsing websites and running background apps.

2. Filtering & Analysis

Used Wireshark filters to inspect specific protocols and activities:

- dns: Analyse domain name lookups

- tcp.analysis.flags: Shows all TCP analysis alerts (e.g., retransmission, duplicate ACKS)

- tcp.stream eq 0: Shows all packets from TCP stream 0 (used to follow conversations)

- tcp.len == 0: Empty TCP payloads-often handshake or keep-alive packets

- tcp.analysis.retransmission: Shows retransmitted packets-may indicate network issues

- tcp.flags.reset == 1: Detect connection resets (RST) – often used in scans/attacks

3. Suspicious Activity Identified

- Repeated DNS queries to unknown domains (possible malware).

- Multiple alerts like duplicate ACKs and retransmissions. May suggest network congestion, packet drops, or interference by a man-in-the-middle (MITM)

- Extended TCP sessions or repeated login attempts. Could point to **credential stuffing**, **brute force**, or **unauthorized access**

- Numerous empty payload packets in rapid succession. May indicate **keep-alive abuse** or **probing behaviour** before attack

- High number of retransmissions from/to specific Ips. Can signal malicious traffic flooding, network instability, or DDoS probing.

- Frequent TCP RST packets across different ports or Ips. Could indicate **port scanning**, **firewall drops**, or **forced connection resets by attacker**

Security Recommendations:

- Implement DNS filtering

- Investigate **network hardware performance**, ensure **IDS/IPS** is not injecting delay, and monitor **host latency**.

- Implement **rate limiting** and **account lockout policies** for login endpoints; enable **2FA** for user accounts.

- Configure **firewall rules** to drop suspicious low-activity keep-alive; verify endpoints for **persistence mechanisms**.

- Audit **network interfaces**, apply **QoS**, and trace heavy outbound flows with **flow monitoring tools**.

- Use **network intrusion detection systems (NIDS)** like Snort/Suricata to flag abnormal port scans and **log RST activity**.

Conclusion:

This analysis using advanced Wireshark filters helped uncover potential security threats such as abnormal DNS activity, TCP retransmissions, and suspicious connection resets. By interpreting these patterns, we identified signs of malware communication, scanning behaviour, and possible brute-force attempts. Implementing the recommended security measures will strengthen network defences and improve overall visibility into malicious traffic.